Handling Exceptions - 0

Fill in the blanks for the definition of java.lang.Error class:

public class java.lang.Error extends _____ {...}

A - Exception

B - Throwable

C - RuntimeException

Handling Exceptions - 1

Which of the following is a checked Exception?

A - ExceptionininitializerError

B - FileNotFoundException

C - RuntimeException

D - ClassCastException

Handling Exceptions - 2

What will be the result of compiling and executing Test class?

```
import java.io.FileNotFoundException;
import java.io.IOException;

abstract class Super {
     public abstract void m1() throws IOException;
}

class Sub extends Super {
     @Override
     public void m1() throws IOException {
         throw new FileNotFoundException();
     }
}

public class Test {
     public static void main(String[] args) {
         Super s = new Sub();
         try {
             s.m1();
         } catch (FileNotFoundException e) {
             System.out.print("M");
         } finally {
             System.out.print("N");
         }
     }
}
```

A - N

B - NM

C - Program ends abruptly

D - Compilation error

Handling Exceptions - 3

What will be the result of compiling and executing Test class?

```
public class Test {
     public static void main(String[] args) {
         try {
             main(args);
         } catch (Exception ex) {
             System.out.println("CATCH-");
         }
             System.out.println("OUT");
     }
}
```

A - Compilation error

B - None of the System.out.printiIn statements are executed

C - CATCH-OUT

D - OUT

Handling Exceptions - 4

Consider below code:

```
public class Test {
     static {
         System.out.println(1/0);
     }

     public static void main(String[] args) {
         System.out.println("HELLO");
     }
}
```

On execution, does Test class print “HELLO” on to the console?

A - Yes, HELLO is printed on to the console

B - No, HELLO is not printed on the console

Handling Exceptions - 5

What will be the result of compiling and executing Test class?

```
public class Test {
     public static void main(String[] args) {
         m1(); //Line 3
     }

     private static void m1() throws Exception { //Line 6
         System.out.println("NOT THROWING ANY EXCEPTION"); //Line 7
     }
}
```

A - NOT THROWING ANY EXCEPTION

B - Compilation error at Line 6

C - Compilation error at Line 3

D - Compilation error at Line 7

Handling Exceptions - 6

What will be the result of compiling and executing Test class?

```
public class Test {
     private static void m1() throws Exception {
         throw new Exception();
     }

     public static void main(String[] args) {
         try {
             m1();
         } finally {
             System.out.println("A");
         }
     }
}
```

A - A is printed to the console, stack trace is printed and then program ends normally.

B - Compilation error.

C - A is printed to the console, stack trace is printed and then program ends abruptly.

D - A is printed to the console and program ends normally.

Handling Exceptions - 7

Which of the following keywords is used to manually throw an exception?

A - throw

B - catch

C - throws

D - thrown

Handling Exceptions - 8

Which of the following are Java Exception classes? Select 3 options.

A - NumberFormatException

B - IllegalArgumentException

C - ClassCastException

D - NullException

E - ArrayIndexException

Handling Exceptions - 9

Consider the following interface declaration:

```
public interface I1 {
    void m1() throws java.io.IOException;
}
```

Which of the following incorrectly implements interface I1?

A -

```
public class C1 implements I1 {
  public void mi() {}
}
```

B -

```
public class C3 implements I1 {
  public void m1() throws java.io. I0Exception{}
}
```

C -

```
public class C4 implements I1 {
  public void m1() throws Exception{}
}
```

D -

```
public class C2 implements I1 {
  public void m1() throws java.io.FileNotFoundException{}
}
```

Handling Exceptions - 10

What will be the result of compiling and executing the following program?

```
import java.io.FileNotFoundException;
import java.io.IOException;

abstract class Super {
     public abstract void m1() throws IOException;
}

class Sub extends Super {
     @Override
     public void m1() throws IOException {
         throw new FileNotFoundException();
     }
}

public class Test {
     public static void main(String[] args) {
         Super s = new Sub();
         try {
             s.m1();
         } catch (IOException e) {
             System.out.print("A");
         } catch(FileNotFoundException e) {
             System.out.print("B");
         } finally {
             System.out.print("C");
        }
     }
}
```

A - AC

B - BC

C - class Sub gives compilation error

D - class Test gives compilation error

Handling Exceptions - 11

What will be the result of compiling and executing Test class?

```
public class Test {
     private static void m1() {
         System.out.println(1/0);
     }

     public static void main(String[] args) {
         try {
             m1();
         } finally {
             System.out.println("A");
         }
     }
}
```

A - A is printed to the console, stack trace is printed and then program ends abruptly.

B - Compilation error.

C - A is printed to the console and program ends normally.

D - A is printed to the console, stack trace is printed and then program ends normally.

Handling Exceptions - 12

Consider below code:

```
public class Test {
     static Double d1;
     static int x = d1.intValue();

     public static void main(String[] args) {
         System.out.println("HELLO");
     }
}
```

On execution, does Test class print “HELLO” on to the console?

A - No, HELLO is not printed on to the console

B - Yes, HELLO is printed on to the console

Handling Exceptions - 13

```
class TestException extends Exception {
     public TestException() {
         super();
     }

     public TestException(String s) {
         super(s);
     }
}

public class Test {
     public void m1() throws __________ {
         throw new TestException();
     }
}
```

For the above code, fill in the blank with one option.

A - Object

B - Error

C - RuntimeException

D - Exception

Handling Exceptions - 14

What will be the result of compiling and executing Test class?

```java
//Test.java
import java.io.FileNotFoundException;

public class Test {
     public static void main(String[] args) {
         try {
             System.out.println(1);
         } catch (NullPointerException ex) {
             System.out.println("ONE");
         } catch (FileNotFoundException ex) {
             System.out.println("TWO");
         }
         System.out.println("THREE");
     }
}
```

A -

```
ONE
THREE
```

B -

```
TWO
THREE
```

C - None of the System.out.printiIn statements are executed

D - THREE

E - Compilation error

Handling Exceptions - 15

What will be the result of compiling and executing Test class?

```
class Base {
     public void m1() throws NullPointerException {
         System.out.println("Base: m1()");
     }
}

class Derived extends Base {
     public void m1() throws RuntimeException {
         System.out.println("Derived: m1()");
     }
}

public class Test {
     public static void main(String[] args) {
         Base obj = new Derived();
         obj.m1();
     }
}
```

A - Derived: m1()

B - Base: m1()

C - Compilation error in Test class

D - Compilation error in Derived class

Handling Exceptions - 16

What will be the result of compiling and executing Test class?

```
public class Test {
    public static void main(String[] args) {
        Error obj = new Error();
        boolean flag1 = obj instanceof RuntimeException; //Line n1
        boolean flag2 = obj instanceof Exception; //Line n2
        boolean flag3 = obj instanceof Error; //Line n3
        boolean flag4 = obj instanceof Throwable; //Line n4
        System.out.println(flag1 + ":" + flag2 + ":" + flag3 + ":" +
         flag4);
    }
}
```

A - false:false:true:false

B - Compilation error

C - false:false:true:true

D - true:true:true:true

E - false:true:true:true

Handling Exceptions - 17

What will be the result of compiling and executing Test class?

```
import java.io.FileNotFoundException;
import java.io.IOException;

abstract class Super {
     public abstract void m1() throws IOException;
}

class Sub extends Super {
     @Override
     public void m1() throws IOException {
         throw new FileNotFoundException();
     }
}

public class Test {
     public static void main(String[] args) {
         Super s = new Sub();
         try {
             s.m1();
         } catch (FileNotFoundException e) {
             System.out.print("X");
         } catch (IOException e) {
             System.out.print("Y");
         } finally {
             System.out.print("Z");
         }
     }
}
```

A - XYZ

B - YZ

C - Compilation Error

D - XZ

Handling Exceptions - 18

Consider below code:

```
public class Test {
    public static void main(String[] args) {
        StringBuilder sb = new StringBuilder();
        try {
            for(;;) {
                sb.append("OCA");
            }
        } catch(Exception e) {
            System.out.println("Exception!!!");
        }
        System.out.println("Main ends!!!");
    }
}
```

What will be the result of compiling and executing Test class?

A - “Exception!!!” is printed on to the console and program terminates abruptly

B - “Main ends!!!” is printed on to the console and program terminates successfully

C - “Exception!!!” is printed on to the console and program terminates successfully

D - “Exception!!!” and “Main ends!!!” are printed on to the console and program terminates successfully

E - Program terminates abruptly

Handling Exceptions - 19

Given Code:

```
import java.io.*;

class ReadTheFile {
     static void print() { //Line 4
         throw new IOException(); //Line 5
     }
}

public class Test {
     public static void main(String[] args) { //Line 10
         ReadTheFile.print(); //Line 11
         //Line 12
     }
}
```

Which 2 changes are necessary so that code compiles successfully?

A - Replace Line 4 with static void print() throws Exception {

B - Surround Line 11 with below try-catch block:

```
try {
  ReadTheFile.print();
} catch(Exception e) {
  e.printStackTrace();
}
```

C - Replace Line 4 with static void print() throws Throwable {

D - Surround Line 11 with below try-catch block:

```
try {
  ReadTheFile.print();
} catch(IOException | Exception e) {
  e.printStackTrace();
}
```

E - Surround Line 11 with below try-catch block:

```
try {
  ReadTheFile.print();
} catch(IOException e) {
  e.printStackTrace();
}
```

F - Replace Line 10 with public static void main(String[] args) throws IOException {

Handling Exceptions - 20

```
public class Test {
     private static int [] arr;
     public static void main(String [] args) {
         if(arr.length > 0 && arr != null) {
             System.out.println(arr[0]);
         }
     }
}
```

Predict Output, if the above code is run with given command?

java Test

A - ArrayIndexOutOfBoundsException is thrown at runtime

B - NullPointerException is thrown at runtime

C - No Output

D - Compilation error

Handling Exceptions - 21

What will be the result of compiling and executing Test class?

```
public class Test {
     private static String s;
     public static void main(String[] args) {
         try {
             System.out.println(s.length());
         } catch(NullPointerException | RuntimeException ex) {
             System.out.println("DONE");
         }
     }
}
```

A - Executes successfully but no output

B - None of the above

C - Compilation error

D - DONE

Handling Exceptions - 22

Consider codes of 3 java files:

```
//Class1.java
package com.training.oca;

import java.io.FileNotFoundException;

public class Class1 {
     public void read() throws FileNotFoundException {}
}
//Class2.java
package com.training.oca;

public class Class2 {
     String Class2;
     public void Class2() {}
}
//Class3.java
package com.training.oca;

public class Class3 {
     private void print() {
         private String msg = "HELLO";
         System.out.println(msg);
     }
}
```

Which of the following statement is true?

A - Only Class3.java compiles successfully

B - Only Class2.java compiles successfully

C - Only Class1.java compiles successfully

D - Class1.java and Class3.java compile successfully

E - Class2.java and Class3.java compile successfully

F - Class1.java and Class2.java compile successfully

Handling Exceptions - 23

Consider below code:

```
public class Test {
     static Double d1;
     int x = d1.intValue();

     public static void main(String[] args) {
         System.out.println("HELLO");
     }
}
```

On execution, does Test class print “HELLO” on to the console?

A - No, HELLO is not printed on to the console

B - Yes, HELLO is printed on to the console

Handling Exceptions - 24

What will be the result of compiling and executing Test class?

```
public class Test {
     private static String s;
     public static void main(String[] args) {
         try {
             System.out.println(s.length());
         } catch(NullPointerException | RuntimeException ex) {
             System.out.println("DONE");
         }
     }
}
```

A - Executes successfully but no output

B - None of the above

C - Compilation error

D - DONE

Handling Exceptions - 25

Consider codes of 3 java files:

```
//Class1.java
package com.training.oca;

import java.io.FileNotFoundException;

public class Class1 {
     public void read() throws FileNotFoundException {}
}
//Class2.java
package com.training.oca;

public class Class2 {
     String Class2;
     public void Class2() {}
}
//Class3.java
package com.training.oca;

public class Class3 {
     private void print() {
         private String msg = "HELLO";
         System.out.println(msg);
     }
}
```

Which of the following statement is true?

A - Only Class3.java compiles successfully

B - Only Class2.java compiles successfully

C - Only Class1.java compiles successfully

D - Class1.java and Class3.java compile successfully

E - Class2.java and Class3.java compile successfully

F - Class1.java and Class2.java compile successfully

Handling Exceptions - 26

Consider below code:

```
public class Test {
     static Double d1;
     int x = d1.intValue();

     public static void main(String[] args) {
         System.out.println("HELLO");
     }
}
```

On execution, does Test class print “HELLO” on to the console?

A - No, HELLO is not printed on to the console

B - Yes, HELLO is printed on to the console

Handling Exceptions - 27

Given code of Test.java file:

```
interface ILogger {
    void log();
}

public class Test {
    public static void main(String[] args) {
        ILogger [] loggers = new ILogger[2]; //Line n1
        for(ILogger logger : loggers)
            logger.log(); //Line n2
    }
}
```

What will be the result of compiling and executing Test class?

A - An exception is thrown at runtime

B - No output is displayed but program terminates successfully

C - Line n2 causes compilation error

D - Line n1 causes compilation error

Handling Exceptions - 28

Given code of Test.java file:

```
abstract class Animal {
    abstract void jump() throws RuntimeException;
}

class Deer extends Animal {
    void jump() { //Line n1
        System.out.println("DEER JUMPS");
    }

    void jump(int i) {
        System.out.println("DEER JUMPS TO " + i + " FEET");
    }
}

public class Test {
    public static void main(String[] args) {
        Animal animal = new Deer();
        ((Deer)animal).jump(); //Line n2
        ((Deer)animal).jump(5); //Line n3
    }
}
```

What will be the result of compiling and executing Test class?

A - Line n2 causes compilation error

B - Line n1 causes compilation error

C - An exception is thrown at runtime

D - Test class executes successfully and prints: DEER JUMPS DEER JUMPS TO 5 FEET

E - Line n3 causes compilation error

Handling Exceptions - 29

Given code of Test.java file:

```
import java.sql.SQLException;

public class Test {
    private static void m() throws SQLException {
        try {
            throw new SQLException();
        } catch (Exception e) {
            throw e;
        }
    }

    public static void main(String[] args) {
        try {
            m();
        } catch(SQLException e) {
            System.out.println("CAUGHT SUCCESSFULLY");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Program ends abruptly

B - Method main(String []) causes compilation error

C - Method m() causes compilation error

D - CAUGHT SUCCESSFULLY is printed on to the console and program terminates successfully

Handling Exceptions - 30

Given code of Test.java file:

```
import java.io.FileNotFoundException;

public class Test {
    static String [] names = {"Williamson.pdf", "Finch.pdf",
        "Kohli.pdf", "Morgan.pdf"};
    public static void main(String[] args) {
        try {
            if (search("virat.pdf"))
                System.out.println("FOUND");

        } catch(FileNotFoundException ex) {
            System.out.println("NOT FOUND");
        }
    }

    private static boolean search(String name) throws
        FileNotFoundException {
        for(int i = 0; i <= 4; i++) {
            if (names[i].equalsIgnoreCase(name)) {
                return true;
            }
        }
        throw new FileNotFoundException();
    }
}
```

What will be the result of compiling and executing Test class?

A - NOT FOUND

B - FOUND

C - Compilation error

D - None of the other options

Handling Exceptions - 31

Given code of Test.java file:

```
import java.sql.SQLException;

public class Test {
    private static void availableSeats() throws SQLException {
        throw null; //Line 7
    }

    public static void main(String[] args) {
        try {
            availableSeats(); //Line 12
        } catch(SQLException e) {
            System.out.println("SEATS NOT AVAILABLE");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Line 7 causes compilation failure

B - SEATS NOT AVAILABLE is printed on to the console and program terminates successfully

C - Line 12 causes compilation failure

D - Program ends abruptly

Handling Exceptions - 32

Given code of Test.java file:

```
public class Test {
    private static void div(int i, int j) {
        try {
            System.out.println(i / j);
        } catch(ArithmeticException e) {
            Exception ex = new Exception(e);
            throw ex;
        }
    }
    public static void main(String[] args) {
        try {
            div(5, 0);
        } catch(Exception e) {
            System.out.println("END");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - END is not printed and program terminates abruptlyEND is printed and program terminates successfully

B - END is printed and program terminates successfully

C - Compilation error

D - END is printed and program terminates abruptly

Handling Exceptions - 33

Given code of Test.java file:

```
public class Test {
    private static void div() {
        System.out.println(1/0);
    }

    public static void main(String[] args) {
        try {
            div();
        } finally {
            System.out.println("FINALLY");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - FINALLY is printed to the console, stack trace is printed and then program ends normally

B - Compilation error

C - FINALLY is printed to the console and program ends normally

D - FINALLY is printed to the console, stack trace is printed and then program ends abruptly

Handling Exceptions - 34

Consider below code snippet available in the same package:

```
abstract class Traveller {
    void travel(String place){}
}

abstract class BeachTraveller extends Traveller {
    /*INSERT*/
}
```

Which of the following declarations/definitions can replace /*INSERT*/ such that there is no compilation error?

Select ALL that apply.

A - void travel(String beach) throws java.io.1OException {}

B - public void travel() throws RuntimeException {}

C - public abstract void travel();

D - public void travel(String beach) throws Exception {}

E - public void travel(Object obj) {}

F - abstract void travel();

G - abstract void travel(String beach);

Handling Exceptions - 35

Given code of Test.java file:

```
import java.io.IOException;
import java.sql.SQLException;

public class Test {
    public static void main(String[] args) {
        /*INSERT*/
    }

    private static void save() throws IOException {}

    private static void log() throws SQLException {}
}
```

Which of the block of codes can be used to replace /*INSERT*/ such that there is no compilation error?

Select ALL that apply.

A -

```
try {
  save();
  log();
} catch(SQLException | |OException ex) {}
```

B -

```
try {
  save();
  log();
} catch(SQLException | Exception ex) {}
```

C -

```
try {
  save();
  log();
} catch(Exception | RuntimeException ex) {}
```

D -

```
try {
  save();
  log();
} catch(lIOException | Exception ex) {}
```

E -

```
try {
  save();
```

```
  log();
} catch(IOException | SQLException ex) {}
```

F-

```
try {
  save();
  log();
} catch(Exception ex) {}
```

Handling Exceptions - 36

Given code of Test.java file:

```
public class Test {
    public static void main(String[] args) {
        try {
            try {
                System.out.println(args[1]); //Line n1
            } catch(RuntimeException e) {
                System.out.print("INHALE-"); //Line n2
                throw e; //Line n3
            } finally {
                System.out.print("EXHALE-"); //Line n4
            }
        } catch(RuntimeException e) {
            System.out.print("INHALE-"); //Line n5
        } finally {
            System.out.print("EXHALE"); //Line n6
        }
    }
}
```

And the commands:

javac Test.java

java Test

What is the result?

A - INHALE-EXHALE-

B - INHALE-EXHALE-INHALE-

C - INHALE-EXHALE

D - INHALE-EXHALE-EXHALE

E - INHALE-EXHALE-INHALE-EXHALE

Handling Exceptions - 37

Given code of Test.java file:

```
import java.io.IOException;

class Parent {
    Parent() throws IOException {
        System.out.print("HAKUNA");
    }
}

class Child extends Parent {
    Child() throws Exception {
        System.out.println("MATATA");
    }
}

public class Test {
    public static void main(String[] args) throws Exception {
        new Child();
    }
}
```

What will be the result of compiling and executing Test class?

A - Test class executes successfully and prints HAKUNAMATATA on to the console

B - Compilation error only in Child class

C - Compilation error only in Parent class

D - Compilation error in both Parent and Child classes

E - Test class executes successfully and prints MATATAHAKUNA on to the console

Handling Exceptions - 38

java.sql.SQLException extends java.lang.Exception

and

java.sql.SQLWarning extends java.sql.SQLException

Given code of Test.java file:

```
import java.sql.*;

interface Multiplier {
    void multiply(int... x) throws SQLException;
}

class Calculator implements Multiplier {
    public void multiply(int... x) throws /*INSERT*/ {

    }
}
public class Test {
    public static void main(String[] args) {
        try {
            Multiplier obj = new Calculator(); //Line n1
            obj.multiply(1, 2, 3);
        } catch(SQLException e) {
            System.out.println(e);
        }
    }
}
```

Which of the options can be used to replace /*INSERT*/ such that there is no compilation error?

Select ALL that apply.

A - RuntimeException

B - NullPointerException

C - SQLException

D - SQLWarning

E - Throwable

F - java.io.IOException

G - Error

H - Exception

Handling Exceptions - 39

Given code of Test.java file:

```
import java.sql.SQLException;

public class Test {
    private static void checkData() throws SQLException {
        try {
            throw new SQLException();
        } catch (Exception e) {
            e = null; //Line 10
            throw e; //Line 11
        }
    }

    public static void main(String[] args) {
        try {
            checkData(); //Line 17
        } catch(SQLException e) {
            System.out.println("NOT AVAILABLE");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - NOT AVAILABLE is printed on to the console and program terminates successfully

B - Line 17 causes compilation failure

C - Line 10 causes compilation failure

D - Line 11 causes compilation failure

E - Program ends abruptly

Handling Exceptions - 40

Given code of Test.java file:

```
public class Test {
    private static void div(int i, int j) {
        try {
            System.out.println(i / j);
        } catch(ArithmeticException e) {
            throw (RuntimeException)e;
        }
    }

    public static void main(String[] args) {
        try {
            div(5, 0);
        } catch(ArithmeticException e) {
            System.out.println("AE");
        } catch(RuntimeException e) {
            System.out.println("RE");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - AE is printed on to the console and program terminates successfully

B - Program ends abruptly

C - RE is printed on to the console and program terminates successfully

D - Compilation error

Handling Exceptions - 41

Given code of Test.java file:

```
public class Test {
    public static void main(String[] args) {
        try { //outer
            try { //inner
                System.out.println(1/0);
            } catch(ArithmeticException e) {
                System.out.println("INNER");
            } finally {
                System.out.println("FINALLY 1");
            }
        } catch(ArithmeticException e) {
            System.out.println("OUTER");
        } finally {
            System.out.println("FINALLY 2");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - OUTER FINALLY 2

B - INNER FINALLY 2

C - INNER FINALLY 1 FINALLY 2

D - INNER FINALLY 1

Handling Exceptions - 42

Given code of Test.java file:

```
import java.io.IOException;

class Super {
    Super() throws RuntimeException {
        System.out.print("CARPE ");
    }
}

class Sub extends Super {
    Sub() throws IOException {
        System.out.print("DIEM ");
    }
}

public class Test {
    public static void main(String[] args) throws Exception {
        new Sub();
    }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error in both Super and Sub classes

B - Test class executes successfully and prints DIEM CARPE on to the console

C - Compilation error only in Super class

D - Test class executes successfully and prints CARPE DIEM on to the console

E - Compilation error only in Sub class

Handling Exceptions - 43

Given code of Test.java file:

```
interface Blogger {
    default void blog() throws Exception {
        System.out.println("GENERIC");
    }
}

class TravelBlogger implements Blogger {
    public void blog() {
        System.out.println("TRAVEL");
    }
}

public class Test {
    public static void main(String[] args) {
        Blogger blogger = new TravelBlogger(); //Line n1
        ((TravelBlogger)blogger).blog(); //Line n2
    }
}
```

What will be the result of compiling and executing Test class?

A - TRAVEL is printed on to the console and program terminates successfully

B - An exception is thrown at runtime

C - Compilation error in TravelBlogger class

D - GENERIC is printed on to the console and program terminates successfully

E - Compilation error in Test class

Handling Exceptions - 44

Given code of Test.java file:

```
import java.sql.SQLException;

public class Test {
    private static void getReport() throws SQLException {
        try {
            throw new SQLException();
        } catch (Exception e) {
            throw null; //Line 10
        }
    }

    public static void main(String[] args) {
        try {
            getReport(); //Line 16
        } catch(SQLException e) {
            System.out.println("REPORT ERROR");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Program ends abruptly

B - REPORT ERROR is printed on to the console and program terminates successfully

C - Line 16 causes compilation failure

D - Line 10 causes compilation failure

Handling Exceptions - 45

Given code of Test.java file:

```
public class Test {
    public static void main(String[] args) {
        try {
            play();
            return;
        } catch(Exception ex) {
            System.out.println(ex.getMessage());
            return;
        } finally {
            System.out.println("MATCH ABANDONED");
        }
        System.out.println("DONE");
    }

    static void play() throws Exception {
        throw new Exception("INJURED");
    }
}
```

What will be the result of compiling and executing Test class?

A - INJURED MATCH ABANDONED

B - INJURED DONE

C - MATCH ABANDONED

D - INJURED MATCH ABANDONED DONE

E - Compilation error

F - INJURED

G - MATCH ABANDONED DONE

Handling Exceptions - 46

Given code of Test.java file:

```
public class Test {
    private static void test() throws Exception {
        throw new Exception();
    }

    public static void main(String [] args) {
        try {
            test();
        } finally {
            System.out.println("GAME ON");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - GAME ON is printed to the console, stack trace is printed and then program ends normally

B - GAME ON is printed to the console, stack trace is printed and then program ends abruptly

C - GAME ON is printed to the console and program ends normally

D - Compilation error

Handling Exceptions - 47

Given code of Test.java file:

```
import java.io.FileNotFoundException;
import java.io.IOException;

class Base {
    Base() throws IOException {
        System.out.print(1);
    }
}

class Derived extends Base {
    Derived() throws FileNotFoundException {
        System.out.print(2);
    }
}

public class Test {
    public static void main(String[] args) throws Exception {
        new Derived();
    }
}
```

What will be the result of compiling and executing Test class?

A - Test class executes successfully and prints 21 on to the console

B - Compilation error only in Derived class

C - Compilation error in both Base and Derived classes

D - Compilation error only in Base class

E - Test class executes successfully and prints 12 on to the console

Handling Exceptions - 48

Given code of Test.java file:

```
class Base {
    public void log() throws NullPointerException {
        System.out.println("Base: log()");
    }
}

class Derived extends Base {
    public void log() throws RuntimeException {
        System.out.println("Derived: log()");
    }
}

public class Test {
    public static void main(String[] args) {
        Base obj = new Derived();
        obj.log();
    }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error in Test class

B - Base: log()

C - Compilation error in Derived class

D - Derived: log()

Handling Exceptions - 49

Given code of Test.java file:

```
public class Test {
    public static void convert(String s)
            throws IllegalArgumentException, RuntimeException,
        Exception {
        if(s.length() == 0) {
            throw new RuntimeException("LENGTH SHOULD BE GREATER
        THAN 0");
        }
    }
    public static void main(String [] args) {
        try {
            convert("");
        }
        catch(IllegalArgumentException | RuntimeException |
        Exception e) { //Line 14
            System.out.println(e.getMessage()); //Line 15
        } //Line 16
        catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

Line 14 causes compilation error. Which of the following changes enables to code to print LENGTH SHOULD BE GREATER THAN 0?

A - Comment out Line 14, Line 15 and Line 16

B - Replace Line 14 with ‘catch(RuntimeException | Exception e) {’

C - Replace Line 14 with ‘catch(IllegalArgumentException | RuntimeException e) {’

D - Replace Line 14 with ‘catch(IllegalArgumentException | Exception e) {’

E - Replace Line 14 with ‘catch(RuntimeException e) {’

Handling Exceptions - 50

Consider below code fragment:

```
import java.util.*;

class A{}
class B extends A{}

abstract class Super {
    abstract List<A> get() throws IndexOutOfBoundsException;
}

abstract class Sub extends Super {
    /*INSERT*/
}
```

Which of the following options replaces /*INSERT*/ such that there is no compilation error?

A - abstract ArrayList **get();**

B - abstract List get();

C - abstract List get() throws ArrayIndexOutOfBoundsException;

D - abstract ArrayList get() throws Exception;

Handling Exceptions - 51

Given code of Test.java file:

```
import java.sql.SQLException;

public class Test {
    private static void getData() throws SQLException {
        try {
            throw new SQLException();
        } catch (Exception e) {
            e = new SQLException();
            throw e;
        }
    }

    public static void main(String[] args) {
        try {
            getData();
        } catch(SQLException e) {
            System.out.println("SQL");
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Method main(String []) causes compilation error

B - SQL is printed on to the console and program terminates successfully

C - Method getData() causes compilation error

D - Program ends abruptly

Handling Exceptions - 52

Given code of Test.java file:

```
public class Test {
    public static void main(String[] args) {
        System.out.println(new RuntimeException()); //Line n1
        System.out.println(new RuntimeException("HELLO")); //Line n2
        System.out.println(new RuntimeException(new
         RuntimeException("HELLO"))); //Line n3
    }
}
```

Does above code compile successfully?

A - No

B - Yes

Handling Exceptions - 53

Given code of Test.java file:

```
import java.io.FileNotFoundException;

public class Test {
    public static void main(String[] args) {
        try {
            System.out.println(args[1].length());
        } catch (RuntimeException ex) {
            System.out.println("ONE");
        } catch (FileNotFoundException ex) {
            System.out.println("TWO");
        }
        System.out.println("THREE");
    }
}
```

What will be the result of compiling and executing Test class?

A - ONE THREE

B - THREE

C - None of the System.out.printiIn statements is executed

D - TWO THREE

E - Compilation error